# HARRY POTTER CONTRE LA FORCE OBSCURE


Harry Potter


La force obscure


Troll


Bâtiment


## Le problème

Trouver le chemin le plus court

de  à 

en évitant les  et les 

*Trouver le chemin le plus court entre deux objets, en évitant les obstacles.*



## Concepts clés de cette mission

- algorithme
- opérations primitives
- conditions
- test *if* (SI … ALORS … SINON)
- boucle *while* (TANT QUE … FAIRE …)
- sous-problème

# Algorithme

Un *algorithme* est une description d'une procédure systématique décrivant des instructions à exécuter.

*Trouver le chemin le plus court entre deux objets, en évitant les obstacles.*



# Opérations primitives

MOVE()

TURN_RIGHT()

TURN_LEFT()

DESTROY_DARK_FORCE ()



# Conditions

CAN_MOVE()

IS_IN_FRONT_OF_ENEMY()

IS_ON_TARGET()



Possibilité de combiner des conditions

## Conditions

| Logique | Python |
|---|---|
| condition **et** condition | `cond1 and cond2` |
| condition **ou** condition | `cond1 or cond2` |
| **non** condition | `not cond` |
| Vrai | `True` |
| Faux | `False` |

## Un peu de logique

| | |
|---|---|
| Vrai **et** Vrai | Vrai |
| Vrai **et** Faux | Faux |
| Faux **et** Vrai | Faux |
| Faux **et** Faux | Faux |



## Un peu de logique

| | |
|---|---|
| Vrai **ou** Vrai | Vrai |
| Vrai **ou** Faux | Vrai |
| Faux **ou** Vrai | Vrai |
| Faux **ou** Faux | Faux |

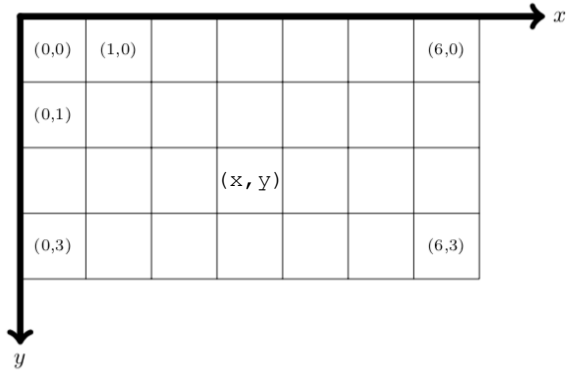


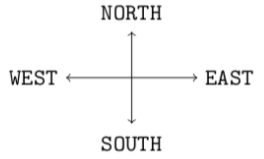
## Un peu de logique

| | |
|---|---|
| **non** Vrai | Faux |
| **non** Faux | Vrai |



## Coordonnées et directions



# Récupération de l'état du terrain

`get_direction()`

Retourne EAST, WEST, SOUTH ou NORTH

`get_x(), get_y()`

Retourne la coordonnée x ou y de 

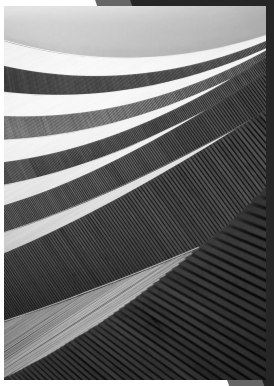
`get_target_x(), get_target_y()`

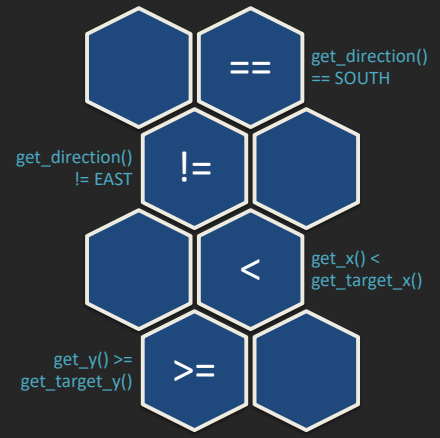
Retourne la position x ou y de 



Conditions

# Opérateurs de comparaison



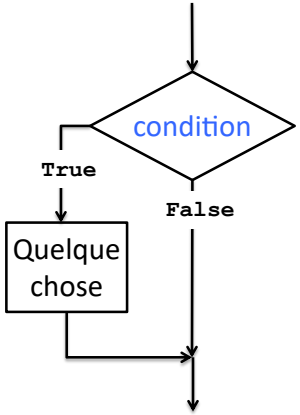





# Test (1 branche)

**SI** condition
**ALORS** Quelque chose

```
if cond:
  qqchose
```

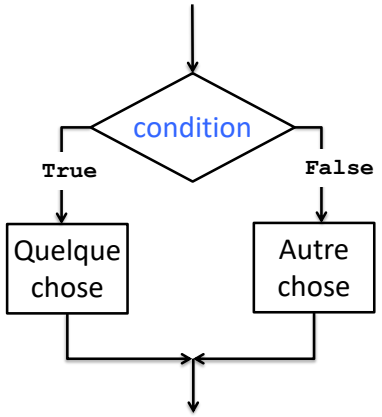





# Test (2 branches)

**SI** condition
**ALORS** Quelque chose
**SINON** Autre chose

```
if cond:
  qqchose
else:
  autrechose
```



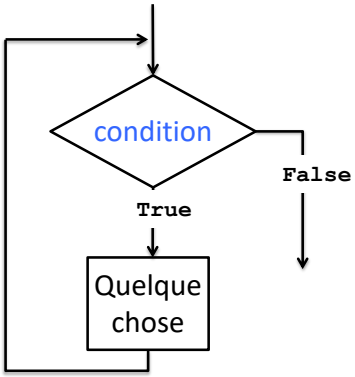
# Exemple d'un test avec 2 branches




# Boucle

**TANT QUE** condition
**FAIRE** Quelque chose

```
while cond:
  qqchose
```






# Exemple d'une boucle

```
TANT QUE  Harry peut bouger FAIRE
|
|     avancer d'une case
|
```

Pseudo code

```
while can_move():

    move()
```

Code



# Combiner une boucle et un test

```
TANT QUE Harry peut bouger FAIRE
|
|     avancer d'une case
|     SI Harry peut bouger ALORS
|     |
|     |     avancer d'une case
|     |
|     SINON
|     |
|     |     tourner vers la gauche
```

Pseudo code

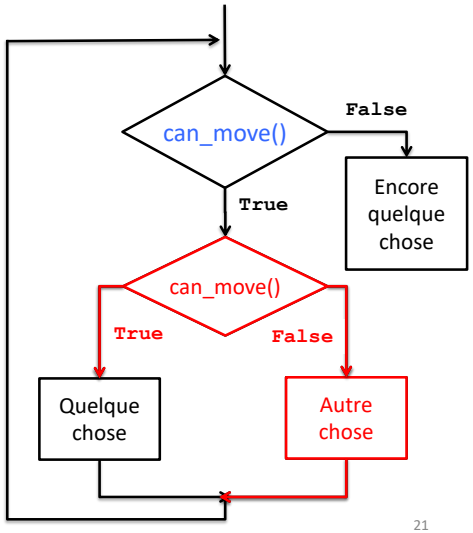
```
while can_move():

    move()
    if can_move():

        move()

    else:

        turn_left()
```

Code

# Combiner une boucle et un test

Quel est le problème avec le programme suivant?

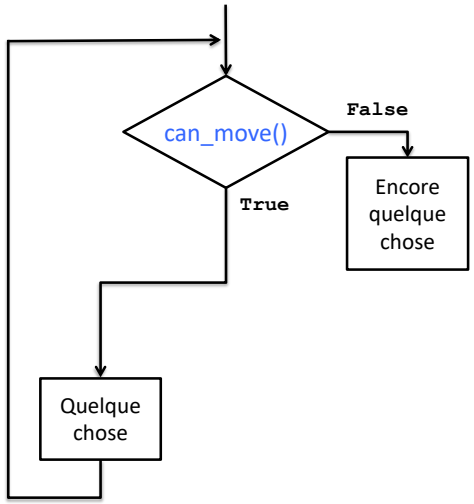
```
while can_move():
    if can_move():
        #qqchose
    else:
        #autrechose
#encoreqqchose
```



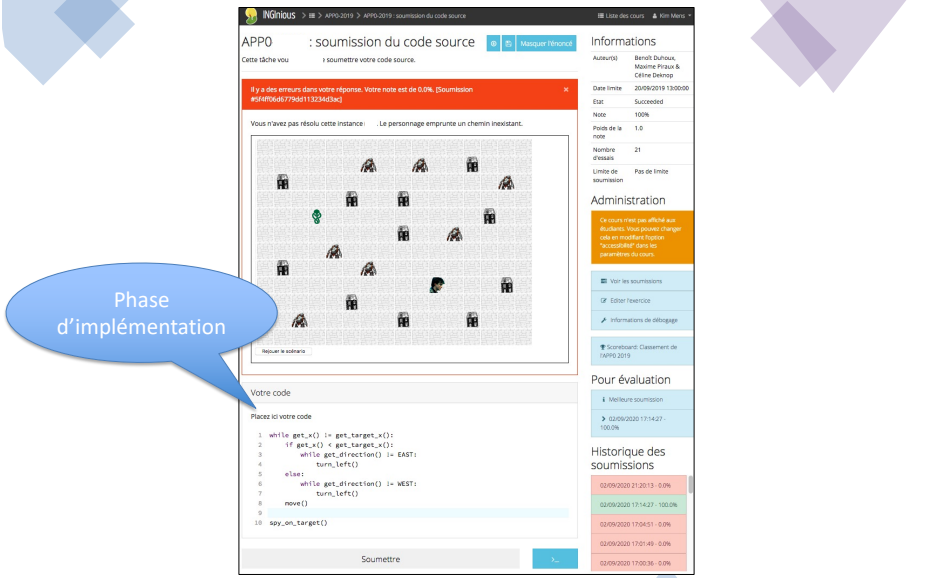



# Combiner une boucle et un test

```
while can_move():

    #qqchose


#encoreqqchose
```

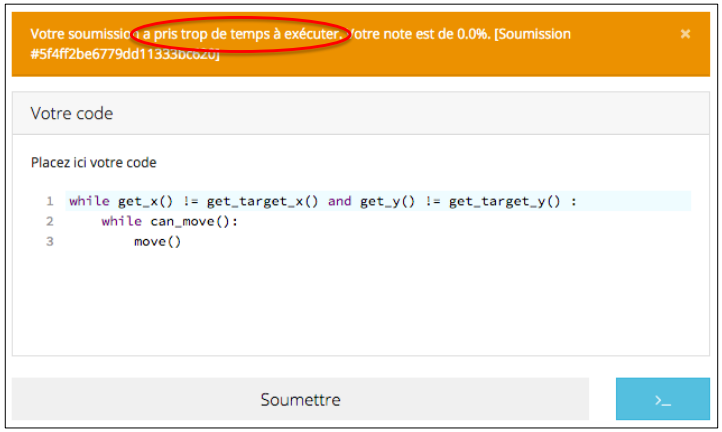




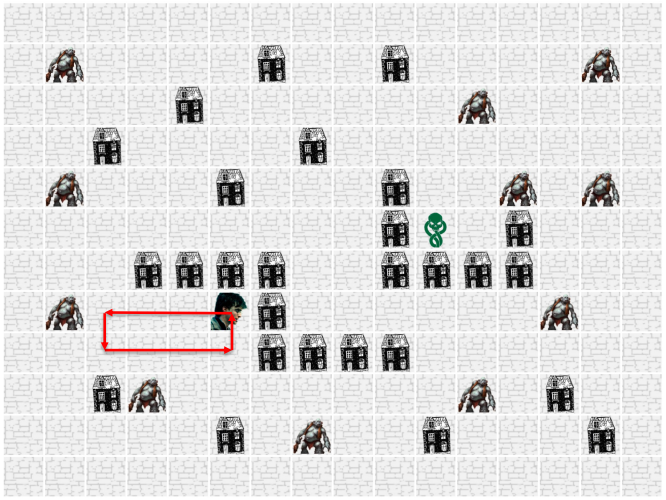






# Attention pour les boucles infinies

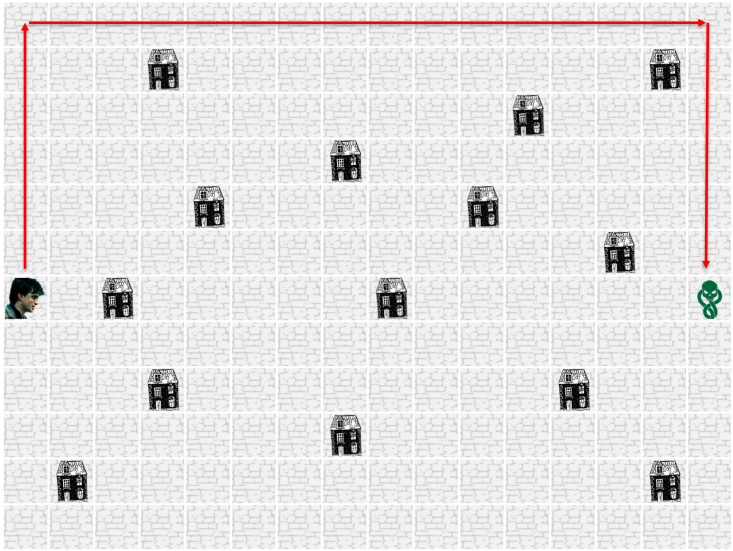


# Attention pour les boucles infinies



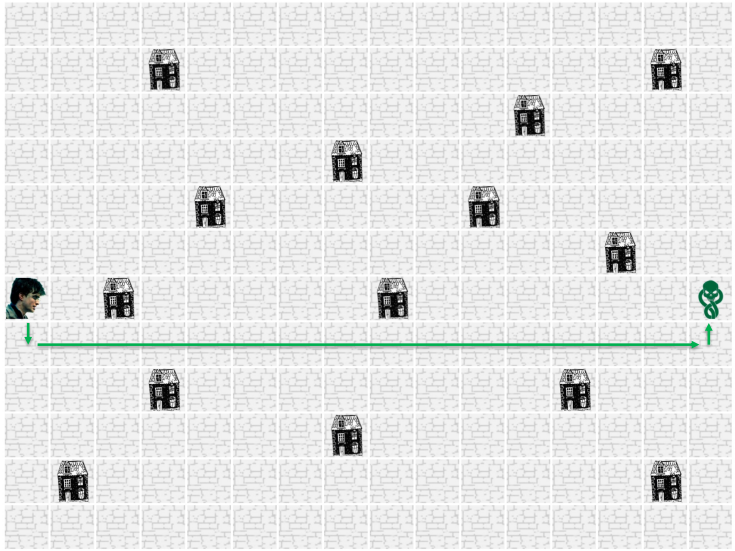


# Trouver le chemin le plus court



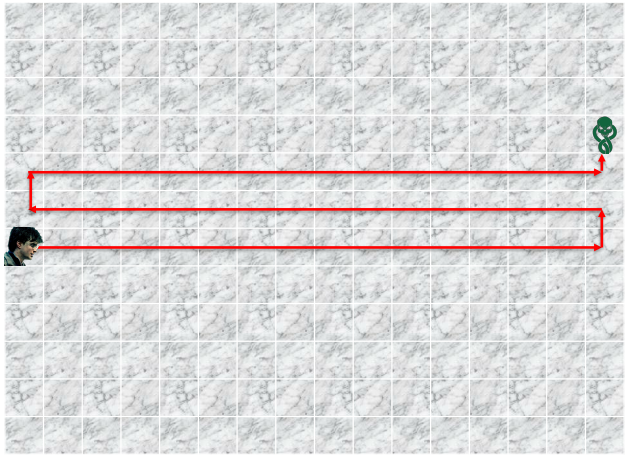


# Trouver le chemin le plus court

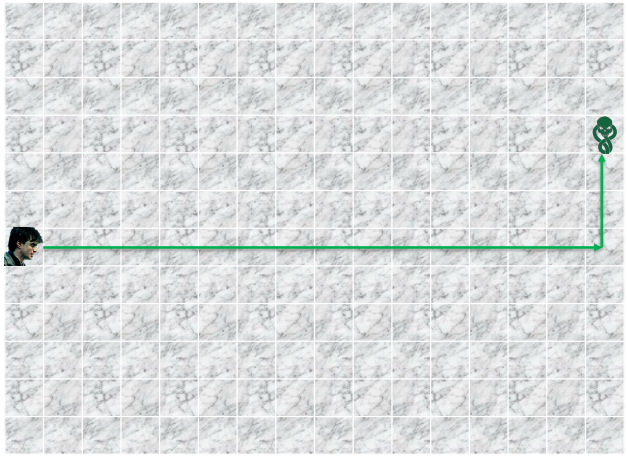




# Trouver le chemin le plus court

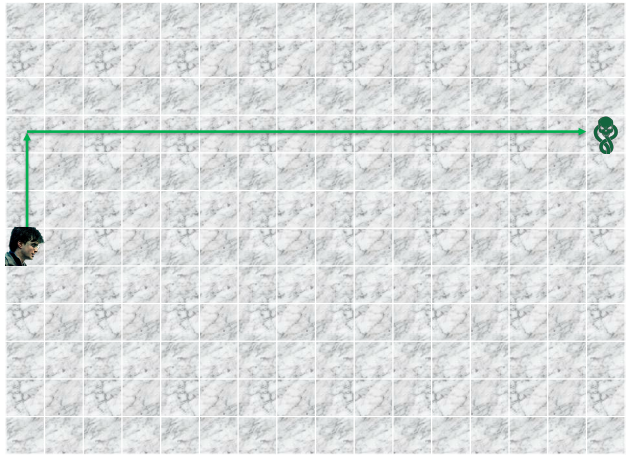


# Trouver le chemin le plus court



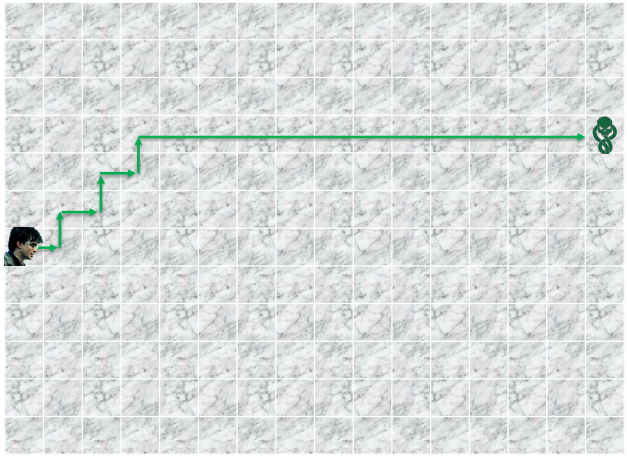


# Trouver le chemin le plus court





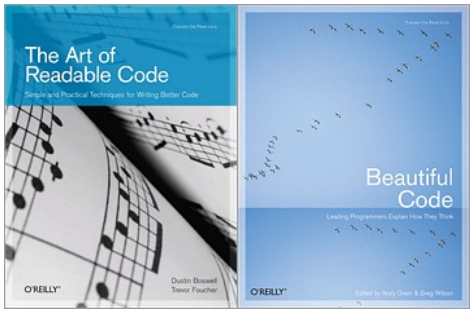
# Trouver le chemin le plus court





# Quel est votre objectif ?

Des programmes
- Ecrits rapidement
- Denses
- Qui utilisent toutes les astuces du langage




Des programmes
- Corrects
  - Le programme doit résoudre correctement le problème posé (même si celui-ci est simple)
- Clairs
  - Le programme doit être lisible et documenté
- Testés
  - Vous devez pouvoir vérifier les limites de votre programme
- Efficaces
  - p.ex. chemin le plus court

# Evitez des programmes trop longs



# Efficacité versus lisibilité

```
while get_direction() != EAST:
    turn_right()
```

Court
Compact
Compréhensible

```
if get_direction == NORTH:
    turn_right()
if get_direction == WEST:
    turn_right()
    turn_right()
if get_direction == SOUTH:
    turn_left()
```

Un peu “plus efficace”
(moins d’operations)

mais beaucoup plus long
et moins compréhensible